The Design of dvitool

Technical Report

S. L. Graham

(415) 642-2059

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Contract No. N00039-82-C-0235

November 15, 1981 – December 30, 1985

Arpa Order No. 4031

†UNIX is a trademark of AT&T Bell Laboratories

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

The Design of dvitool

Jeffrey W. Mc Carrell

Dvitool, a TEX output previewer running on the SUN workstation[1] under their proprietary window system, is an integral part of the Berkeley TEX environment. TEX[5] is a document formatter that outputs a DeVice Independent (DVI) file which contains a low-level, machine-like page description language. Dvitool presents an image of the page on the workstation just as it would appear on the printed page, so it functionally replaces the printer in a write—TEX—print—edit cycle. This paper discusses dvitool's development, current usage, and some future directions.

Keywords: man computer interface; portability.

1 Features

The primary goal for dvitool was to provide a means to view the DVI representation of a TEX file without resorting to a printer. The rest of this section describes dvitool's primary features.

1.1 Initialization

One of the first things dvitool does when run is look for a user specific customization file. The customization file describes initialization parameters which are mostly window system and user interface specific, for example, the placement and size of the window dvitool runs in.

1.2 Reading the DVI Page

After dvitool has initialized itself, it scans backwards through the DVI file and builds a list of pointers to each of the DVI pages. DVI files contain a postamble which has a pointer to the last page. Each page, in turn, has a pointer to the page before it, so building the list of page pointers doesn't require unreasonable amounts of time. For documents of 40 pages or less, the delay is hardly noticable. After the page list is built, dvitool presents the first page.

[1]SUN workstation is a trademark of Sun Microsystems, Inc.

DVI pages consist of a stream of low level typesetting commands[3], such as "set character n from the current font at the current x and y", "move horizontally by x units", and "switch to font f." `Dvitool`'s `read_page` routine iterates from the `BOP` (beginning of page) command to the `EOP` command, painting characters onto an internal buffer. `Dvitool`'s fonts contain the actual raster images of the characters, so character painting is done with raster operations. The arithmetic used to round from TEX's infintessimal scaled points[2] to pixels is done very carefully to insure a faithful representation of the page. After the entire page has been built, the visible portion of the page is painted onto the user's window in one raster operation.

1.3 Page Scaling

The image `dvitool` presents of the DVI page is scaled up 1.45 times. When `dvitool` is made as big as the screen will allow, the full width of the page and about 70% of the height of the page are visible. It would have been preferable not to have scaled the page up at all so the full page would have been visible,[3] but the SUN screen doesn't have enough resolution. At the SUN's resolution (80 dots per inch) a 1.45X magnification is close to the lower bound of usability. More concretely, without the scale facter, a point would map to 1.1 pixels [4]. The longest dimension of characters in ten point fonts would thus have to be represented in $\approx$ 11 pixels. With the 1.45 scale factor, 16 pixels[5] are available in each dimension. Since these are not fixed width fonts, any given character probably is not actually 16 pixels high or wide; 16 pixels is just the upper bound for 10 point fonts at a 1.45 scale factor.

1.4 Intra-page Movement

Once the page is painted, the user can scroll around arbitrary amounts either vertically or horizontally. The default action is to scroll one third of the window size. This works out well enough for horizontal scrolling, but a better scheme for vertical scrolling would be to place the lowest visible line at the top of the window, much like a text editor scrolls. There are also commands to position on any edge of the page, so one keystroke positions the bottom of the page at the bottom of the window. Recall that the complete DVI page has already been read and painted on an internal buffer, so new views of the same page are instantaneous.

---

[2] "points" are dimensions used to describe fonts, kerns and the like. 1 inch = 72.27 points; 1 point = 2^{16} scaled points. Thus 1 inch $\approx$ 4.7 million scaled points.

[3] The SUN screen has 900 vertical pixels and 1100 horizontal pixels. 900/80 dots per inch = 11.25 so a full 8.5 by 11 sheet of paper would be visible with no scale factor.

[4] 80 pixels per inch /72.27 points per inch = 1.106 pixels per point.

[5] $1.45 * 80/72.27 = 1.605$

1.5 Inter-page Movement

To move back and forth across pages, dvitool must read and paint a new image so there is a short delay, typically 4 seconds in our environment. Pages are cached, however, so that once a page has been viewed, viewing it again is nearly instantaneous. The memory penalty for page caching is about 6K bytes per page, which is not too prohibitive for SUN workstations with 4 megabytes of memory. The user can limit the number of cached pages to control thrashing, and dvitool internally sets the limit whenever it cannot obtain enough memory to cache another page.

1.6 Page Searches

The values of TEX's 10 \count variables is stored with each page in the DVI file. Most TEX macro packages store quantities in these 10 variables which correspond to the logical structure of the document, such as chapter or section numbers. The logical (TEX assigned) page number is usually stored in \count0. Dvitool has two notions of page numbers: the logical page number which corresponds to the 10 \count values, and the physical page number which corresponds to the physical order of the pages in the DVI file. In addition to physical page searches, dvitool has "wildcard" logical page searches. The "match anything" character (*) can be used for any of the 10 \count fields. Users who know how to correlate the information in the \count variables can page through the document using it's logical structure, for example, to go to the first page of chapter 4. Dvitool displays the value of all of the \count variables on the status line for each page,[6] so the user can see how the \count variables are used. Commands also exist to view the first page, the last page, or the page at any offset from the current page. The movement commands are reminiscent of a text editor.

1.7 Magnification

Global magnification has been implemented in dvitool. TEX's \magnification macro magnifies all dimensions unless they are specified in "true" dimensions. TEX keeps \hsize and \vsize in true points, so DVI pages are usually 8.5 by 11 inchs, regardless of the \magstep used. Dvitool's magnification, on the other hand, is global. It simply magnifies the entire page. There are 6 steps available, corresponding to TEX's 6 \magsteps. Discrete steps of magnification were implemented rather than a continuous spectrum because additional magnification steps require additional fonts. Dvitool's fonts already require 1.4 megabytes of disk space.

[6] This isn't quite true. Trailing zeros are elided.

1.8 DVI information

Dvitool can also report information about the DVI image, though this capability is limited. DVI files were designed to be a compact representation of a typeset page. There isn't a lot of extraneous information in them, so there isn't much that dvitool can report. Perhaps the most useful piece of reportable information is the font in which a character is set. Even the font name is of limited usefulness, however, because the user has to correlate the font name in the TEX document—which may have gone through arbitrary macro expansion—to dvitool's name for the font. For example, TEX users in our environment have to know that TEX uses amitt for italic fonts obtained with the \it macro. LATEX[6] users have to correlate amitt with emphasized text (\em) as well. This is an example of the more general problem of how to map from one representation of an object to another. Dvitool reports what it can and leaves the user to interpret the information.

2 User Interface

The user interface to dvitool was the subject of a number of experiments. The SUN window environment[1] offers many different ways to invoke commands. We finally decided on two: keystrokes and menus. These were chosen because we found that novice users of dvitool expected to use the mouse to perform commands in a window environment, while advanced users found the menus slow and cumbersome. Dvitool provides both so it is both easy to learn for the novice and responsive to the expert. Clues are provided to aid the user in the transition from novice to expert. For example, all of the pop-up menu entries also contain the matching keystroke command. Another clue we provide is a help facility which is itself a DVI file. The help file teaches users how to use dvitool as they read it. We provide the TEX source for the help DVI file so individual sites can customize the help facility.

Expert users who do not need the visual prompting of menus prefer the faster keystroke commands, so dvitool provides them. Experts can also redefine the key bindings to make dvitool look like their favorite flavor of editor. This feature is particulary important because users frequently switch from an editor containing TEX source to dvitool and back.

We also considered having "buttons" as our primary user interface. Buttons are fixed areas of window real estate that the user points to and clicks on with the mouse to invoke a command. They differ from menus in that menus pop-up; they are visible only when requested. Buttons were rejected in favor of keyboard commands for two reasons: 1) they require screen real estate which is better utilized to display the DVI page; and 2) because buttons are mouse based and most editors are keyboard based, moving from editor to dvitool would require changing from keyboard to mouse. Advanced users find the change of input device slow and annoying. Buttons may find their way into dvitool when mouse

based editors become more readily available.

3 Portability

Dvitool was developed on SUN hardware and runs under their proprietary window system. Some care has been taken to isolate the system dependent parts of the code, but any program which must deal intimately with a non-standardized graphics interface is inherently not very portable. Dvitool is typical in this respect. We expect to begin work on a port to the X window system[4] soon.

4 texdvi

Texdvi is a companion program for dvitool which in one step runs TEX and previews the resulting DVI file with dvitool. Texdvi is smart enough to preview the DVI file only if it was changed by TEX. In addition, if there is a dvitool running, texdvi will send a request to the running dvitool to preview the DVI file rather than start up another dvitool to do the job. Texdvi is a simple but useful means of automating the process of running TEX and previewing the results.

5 Future Directions

Over time, the user interface to dvitool has become more editor-like. Since it is possible, and indeed desirable, to have both a text editor and dvitool on the screen at the same time, a number of dvitool's features are customizable so it can be used with a wide range of text editors. We expect this trend to continue. Currently planned addtions to dvitool include negative magnification (shrinking), better vertical scrolling as described earlier, and a word search facility.

The word seach facility will need to provide some way to map from ASCII to the DVI representation. In particular, ligatures present problems. At the DVI level, ligatures such as the two characters "ff" are a single character. A compile time translation table could do the mapping, but that solution is necessarily dependent on external and potentially changeable information.

Another problem is mathematical text. What pattern would the user type in ASCII to search for x_1, for example? The obvious solution of having dvitool recognize the TEX syntax for that expression implies that dvitool would have to be able to parse the TEX language which is a task far beyond its scope.

A related problem also rises from the problem of mapping ASCII to a larger set of characters. Some of the symbol fonts have non-letter characters (e.g. $\bowtie$) in the same

positions as letters in character fonts. Since matching is done on the character index in the font, this suggests that a naive search facility might incorrectly match ⋈ when the user had requested an ASCII character. We expect that the search facility will simply ignore characters set in unknown fonts. This is an area of ongoing research.

6 Conclusions

Dvitool has grown to be a very useful tool for writing TEXnical documents. It doesn't supplant editing with hard copy and a red pen, but it does allow iterations through the write—TEX—revise cycle without resorting to hard copy. It is expected that much of dvitool's design and code will be used in the display part of VORTEX[2].

References

[1] *SunView Programmer's Guide, Release A of 17.* Sun Microsystems, Mountain View, California, February 1986.

[2] Peehong Chen, John Coker, Michael A. Harrison, Jeffrey W. McCarrell, and Steve Procter. The VORTEX document preparation environment. In Preparation.

[3] David Fuchs. Device independent file format. *TUGBoat*, 3(2):14–19, October 1982.

[4] Jim Gettys and Ron Newman. *Xlib - C Language X Interface: Version 9.* MIT Project Athena, Cambridge, Massachusetts, 1985.

[5] Donald E. Knuth. *The TEX Book.* 1984.

[6] Leslie Lamport. *LATEX: A Document Preparation System. User's Guide and Reference Manual.* 1986.